AD-A054 040 ARINC RESEARCH CORP SANTA ANA CALIF WESTERN DIV F/G 9/2
TACTICAL DATA SYSTEM COMPUTER PROGRAMMING SPECIFICATION,(U)
NOV 66 C W MCINDOE, C KIMME N123(61756)56869A
414-04-4-692 NL

UNCLASSIFIED

| OF
AD
A054040

END
DATE
FILMED
6-78
DDC

(12)

TACTICAL DATA SYSTEM COMPUTER PROGRAMMING SPECIFICATION

November 1966

Prepared for

FLEET COMPUTER PROGRAMMING CENTER, PACIFIC
San Diego, California

Under Contract N123(61756)56869A

Publication No. 414-04-4-692

ARINC RESEARCH CORPORATION

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 414-04-4-692✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* TACTICAL DATA SYSTEM COMPUTER PROGRAMMING SPECIFICATION		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER 414-04-4-692
7. AUTHOR(s) C.W. McIndoe Carolyn Kimme		8. CONTRACT OR GRANT NUMBER(s) N123(61756)56869A✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS ARINC Research Corporation✓ 2551 Riva Road Annapolis, Maryland 21401		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS FLEET COMPUTER PROGRAMMING CENTER, PACIFIC San Diego, California		12. REPORT DATE November 1966 13. NUMBER OF PAGES 35
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* FLEET COMPUTER PROGRAMMING CENTER, PACIFIC San Diego, California		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

UNCLASSIFIED/UNLIMITED

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

(12)

(6) TACTICAL DATA SYSTEM COMPUTER PROGRAMMING SPECIFICATION,

(11) Nov[redacted]66

(12) 67 p.

Prepared for

FLEET COMPUTER PROGRAMMING CENTER, PACIFIC
San Diego, California

Under Contract (15) N123(61756)56869A

Prepared by

C. W. McIndoe
(10) C. W. McIndoe
Carolyn Kimme
Carolyn Kimme

Approved by

pgu Carlson
for W. C. Hanna

ARINC RESEARCH CORPORATION
Western Division
P. O. Box 1375
Santa Ana, California

(14) Publication No. 414-04-4-692

406 547

JQB

RECORD OF CHANGES

CHANGE NO.	DATE	TITLE OR DESCRIPTION	SIGNATURE OF VALIDATING OFFICER

SECURITY INFORMATION — CLASSIFICATION ( )

LIST OF CHANGED PAGES ISSUED

Insert latest changed pages; destroy superseded pages.

Notes: 1. The portion of the text affected by the current change is indicated by a vertical line in the outer margins of the page.

2. The asterisk (*) indicates pages changed, added or deleted by the current change.

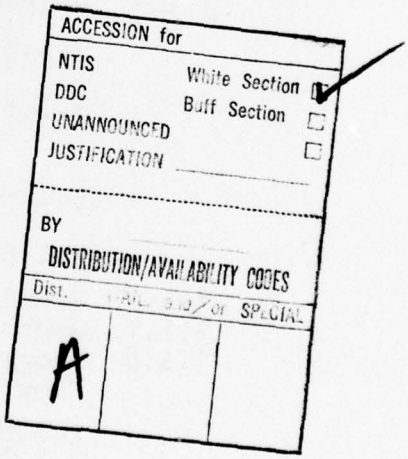


TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

1. SCOPE AND PURPOSE

1.1 GENERAL

1.1.1 This specification defines the requirements for each of the deliverable items of a computer program, including the associated documentation items. This specification also defines the precise relationship between the items that are required for understanding the program and for accessing data within each document.

1.1.2 Additionally, the specification defines the requirements for configuration control of each item to permit tracing of the program evolution through the complete program life cycle. The precise configuration control specified will permit the simultaneous existence of two or more configurations during the evolutionary process. Multiconfiguration is the by-product of a dynamic tactical situation and the constant pressure to improve systems effectiveness.

1.1.3 To accomplish these goals, data items are required to document each process in the program life cycle--from design, through coding and testing, to the permanent documentation required for training of, and field use by, personnel who must be rotated regularly because of other military constraints. Additionally, the data items must be compatible with revision processes that will result inevitably from knowledge gained from Fleet use and from improvements in the state of the art of the equipment deployed in the Fleet.

1.1.4 The required data items are listed subsequently; their contributions to the overall program and some of the reasons for their existence are provided in the following paragraphs.

(a) Document Control List
(b) Program Flow Diagrams
(c) Functional Description of Logic Flow
(d) Functional Specification
(e) Functional Flow Charts
(f) Program Card Deck

(g) Source Program Listing
(h) Assembled Program Listing
(i) Testing Routines
(j) Program Tape

1.2 DOCUMENT CONTROL LISTING

1.2.1 To control the evolutionary process from tactical requirement to operational program, a document control numbering system has been established and is described in Section 3.1.16, Data Management Control Block. This system has a logical structure such that the same types of documents (i.e., operational specifications) will have similar numbers, and all documents of a specific configuration will have related numbers. The evolution of the control numbers must be documented formally and maintained. Toward this end, the requirements for a Document Control List have been established.

1.3 PROGRAM FLOW DIAGRAMS AND FUNCTIONAL DESCRIPTION OF LOGIC FLOW

1.3.1 Program Flow Diagrams and Functional Description of Logic Flow are designed jointly to document the program design portion of the program life cycle. Since the personnel who perform the program design are normally separate from the personnel who perform the coding task, it is necessary that the logic and technique used by the program designer be documented in some permanent fashion. Such a permanent record is mandatory because there may be a considerable time lapse between the design of the program and its final loading and field use; or because of advancement, reassignment within the company, or relocation to a new or different company, the program designer may not be available to explain or defend the design of the program. Toward this end, the illustration of the logic and the functional grouping of logic by use of a Program Flow Diagram, and the narrative description of the parameters surrounding the logic by use of the Functional Description of Logic Flow, were developed.

1.4 FUNCTIONAL SPECIFICATION

1.4.1 Since the personnel who code the program are organizationally and sometimes geographically separate from the personnel who design the program, a vehicle of communication is necessary to bridge the gap. To accomplish this, the Program Flow Diagrams and the Functional Descriptions developed during program design are reviewed, corrected, and finalized at the critical design review at the completion of the Logic Design Phase. Following approval, the Functional Descriptions of Logic Flow are renamed and become the Functional Specification, which is a binding document during the coding of a program. Finally, but no less important, it may not be apparent which portion of the coding corresponds precisely to a specific portion of the logic design. Hence, the subfunctional grouping technique of the Program Flow Diagram is provided as a firm base for reference between not only the coding and the Program Flow Diagram, but between all levels of data items.

1.5 FUNCTIONAL FLOW CHART

1.5.1 The Functional Flow Chart is the programmer's illustration of the technique used to make the computer at hand satisfy the requirements of the program design.

1.5.2 The Functional Flow Charts are similar in design and content to the Program Flow Diagram; they illustrate the programmer's approach to the coding process, rather than the designer's interpretation of the Operational Specification. The flow charts illustrate how the program satisfies the requirements of the design, and the actual sequence of events that occurred in the coding process, rather than the options available as illustrated by the Program Flow Diagram. The Functional Flow Charts will be used for personnel training and for Fleet documentation, while the Program Flow Diagrams shall be maintained as a base for future revisions to the design of the programs.

1.6 PROGRAM CARD DECK AND SOURCE PROGRAM LISTING

1.6.1 These documents record the program coding portion of the program life cycle. Additionally, through the use of identifying numbers on the comment cards, they provide a precise reference to the Program Flow Diagram which documents the portions of the logic design requirements being satisfied by coding. The requirement for identification of the subfunctional breakdown of the coding provides a facility for identification of the portion of the coding that satisfies the logic requirements illustrated on the Program Flow Diagrams.

1.6.2 To accomplish this identification task, the requirements for comment cards was instituted. By use of comment cards which do not affect the compilation or running of the program, the subsections of the card deck and the program listing can be identified and their purpose described.

1.6.3 This identification and logic description is a good general practice for all programming, and specifically desirable for military computer programs. The identification and description is extremely valuable during the debugging process.

1.6.4 With the constant pressure to improve the program generally, and to change the program to meet changing tactical requirements, it is imperative that complete visibility and access be provided to the various portions of the program by the comment cards. With the visibility and access provided, proposed changes can be analyzed effectively for impact on the program, and, as necessary, programs can be updated more easily to meet the tactical requirements.

1.7 TESTING ROUTINES

1.7.1 The requirements for testing are self-evident; some of the associated problems, however, are not. Testing is divided into two categories: (1) Logical Testing (debugging) which is performed to ensure that the program will run; and (2) Simulation Testing which is performed to ensure that the program will meet the mission requirements.

1.7.2 Debugging is an important part of the computer programming process, It is unlikely that the first attempt at running any program will be successful. Since the probability of error is high, the precise documentation technique discussed in the previous paragraphs will be extremely advantageous to the contractor developing the program. While the program is being debugged the Program Card Deck will be changed. It is imperative that the Program Flow Diagrams, Functional Description, comment cards, etc., be updated to reflect these changes.

1.7.3 Logical Testing Routine

1.7.3.1 To ensure that the program will run, Logical Testing Routines which exercise the logic of the program routines must be developed. With these routines available, program routines can be checked out as they are coded, thus reducing the debugging burden near the final delivery milestone.

1.7.4 Simulation Testing Routine

1.7.4.1 To ensure that the program will accomplish its intended task, a testing routine which simulates the operational environment and contains realistic data must be developed. Ultimately the program will be Fleet tested to ensure that a program will meet Fleet requirements.

1.8 SUMMARY

1.8.1 The specification provides precise requirements for the development and control of the necessary programming information associated with computer software production. Each documentation device has been carefully designed to meet a specific need, and to have a specific relationship to the other documentation devices. Quality control requirements have been established and will be adhered to. By use of these documentation techniques and by careful attention to technical accuracy and consistency, satisfactory computer program documentation will result.

2. APPLICABLE DOCUMENTS

2.1 GOVERNMENT DOCUMENTS

2.1.1 The following government document(s), of the issue in effect on date of invitation for bids or request for proposal, form a part of this specification to the extent specified herein.

MIL-STD-682, Flow Chart Symbols for ADP Systems

(Copies of specifications, standards, drawings, and publications required by suppliers in connection with specific procurement functions should be obtained from the procuring activity or as directed by the contracting officer.)

Naval Tactical Data System, CS-1 Manual.

3. REQUIREMENTS

3.1 PROGRAM FLOW DIAGRAMS AND FUNCTIONAL FLOW CHARTS

3.1.1 This subsection specifies the requirements for preparing the graphic representations that illustrate, on a single sheet, the logical flow of data and the sequence of operations in a digital computer program, routine, or subroutine, through the use of geometric shapes, symbols and supplementary notations. Unless otherwise noted, all requirements of this section apply equally to both types of diagrams. The style and format remain the same, only the level of detail varies. The two types of graphic documentation required by this specification are described below.

3.1.2 Program Flow Diagram

3.1.2.1 The Program Flow Diagram (see Figure 1) is the basic document illustrating the decision processes, and resulting actions in terms of design logic. The design logic includes tests such as "target closer than 200 miles", "target displaying IFF", "target confirmed", and actions such as Set; Drop Track Bit; Subtract Range from Previous Range; Compute; New Target Position. When multiple actions are to be performed after a test or series of tests, optional sequences must be identified to provide flexibility in the programming process. With the options illustrated, the programmer can attempt various combinations to improve the efficiency of the program.

3.1.3 Functional Flow Chart

3.1.3.1 A detailed extension of the Program Flow Diagram, the Functional Flow Chart, illustrates the programming process required to satisfy the logic design requirements of the Program Flow Diagram, and the options selected. Figure 2 provides an example of a Functional Flow Chart.

3.1.3.2 In addition to containing blocks for each logical decision and action, the Functional Flow Chart shall contain blocks for

programming operations such as masking, shifting, incrementing, clearing, storing, exclusive oring, etc.

3.1.3.3 The terminology in the decision and action boxes shall reflect the coding of the requirements illustrated on the Program Flow Diagram. For example, where the Program Flow Diagram illustrates "Set Drop Track Bit", the Functional Flow Chart illustrates "Set DT=1" where DT has been defined as the word section or table that contains drop track information.

3.1.3.4 Computer prepared Functional Flow Charts shall comply with the following requirements for content, but shall be exempted from the requirements for format.

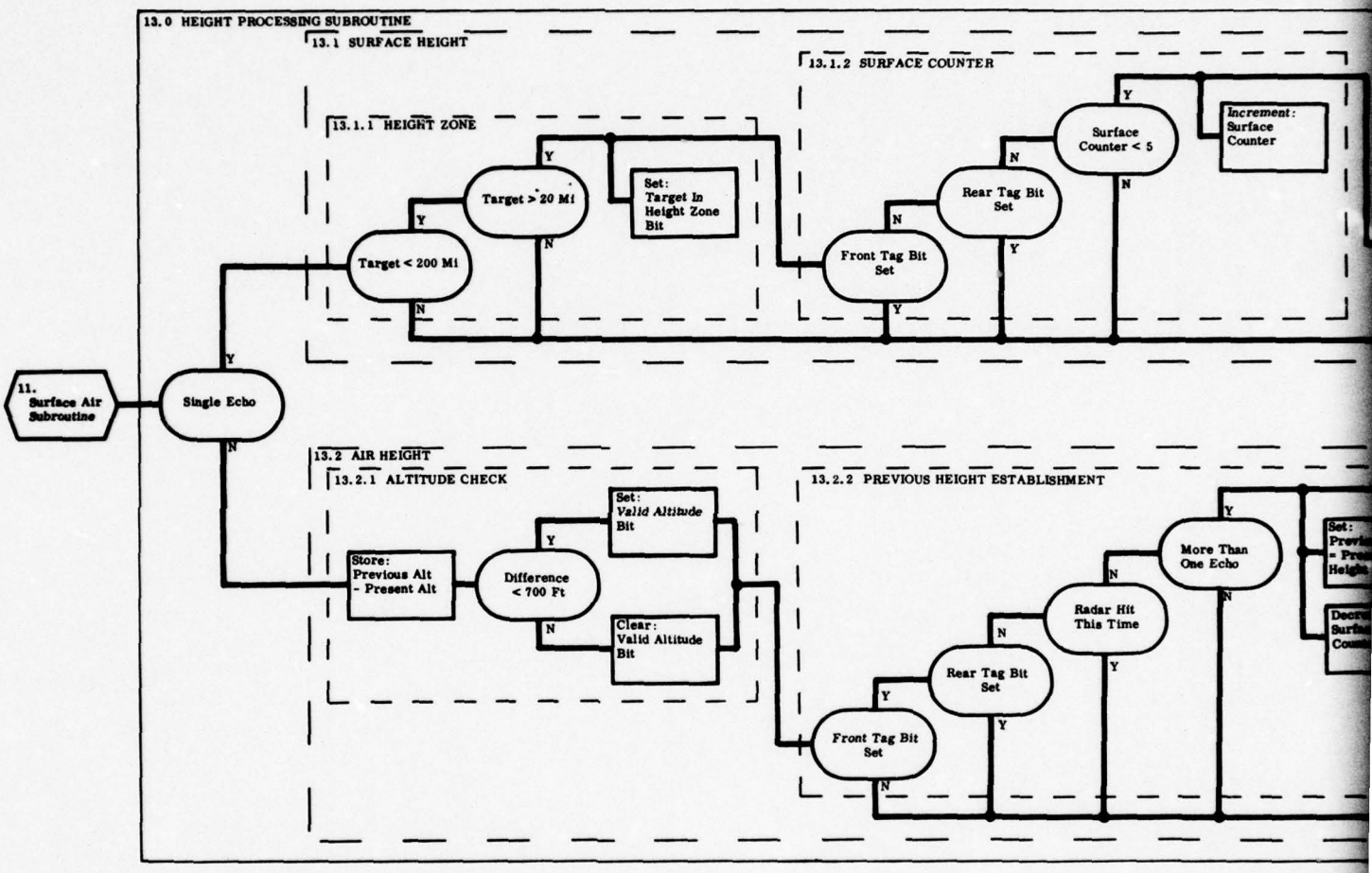
3.1.4 Preparation

3.1.4.1 A Program Flow Diagram and Functional Flow Chart shall be prepared for each of three program subdivisions--one illustrating the division of the complete program into routines, one illustrating the division of a routine into subroutines, and one illustrating the flow within each of the subroutines.

3.1.5 Degree of Detail

3.1.5.1 The degree of detail is influenced by the problem, by the programming language to be used, and by the level of program subdivision being flow-charted. Flow diagrams that deal with low-level program subdivisions require more detail than flow charts that deal with the higher levels.

3.1.5.2 Overall Program Flow Diagram and Overall Functional Flow Chart. The overall Program Flow Diagram and Overall Functional Flow Chart shall contain one block for each major routine. They shall have one block for each peripheral device which provides input to, or receives output from, the computer being programmed. They shall have lines connecting the boxes to illustrate the flow within the program, with separate lines to be used for each basic flow. When direction of flow is not readily apparent, arrowheads shall be used to indicate direction of flow.


13.0 HEIGHT PROCESSING SUBROUTINE
13.1 SURFACE HEIGHT
13.1.1 HEIGHT ZONE
13.1.2 SURFACE COUNTER
11. Surface Air Subroutine
Single Echo
Target < 200 Mi
Target > 20 Mi
Set: Target In Height Zone Bit
Front Tag Bit Set
Rear Tag Bit Set
Surface Counter < 5
Increment: Surface Counter
13.2 AIR HEIGHT
13.2.1 ALTITUDE CHECK
13.2.2 PREVIOUS HEIGHT ESTABLISHMENT
Store: Previous Alt - Present Alt
Difference < 700 Ft
Set: Valid Altitude Bit
Clear: Valid Altitude Bit
Front Tag Bit Set
Rear Tag Bit Set
Radar Hit This Time
More Than One Echo
Y
N

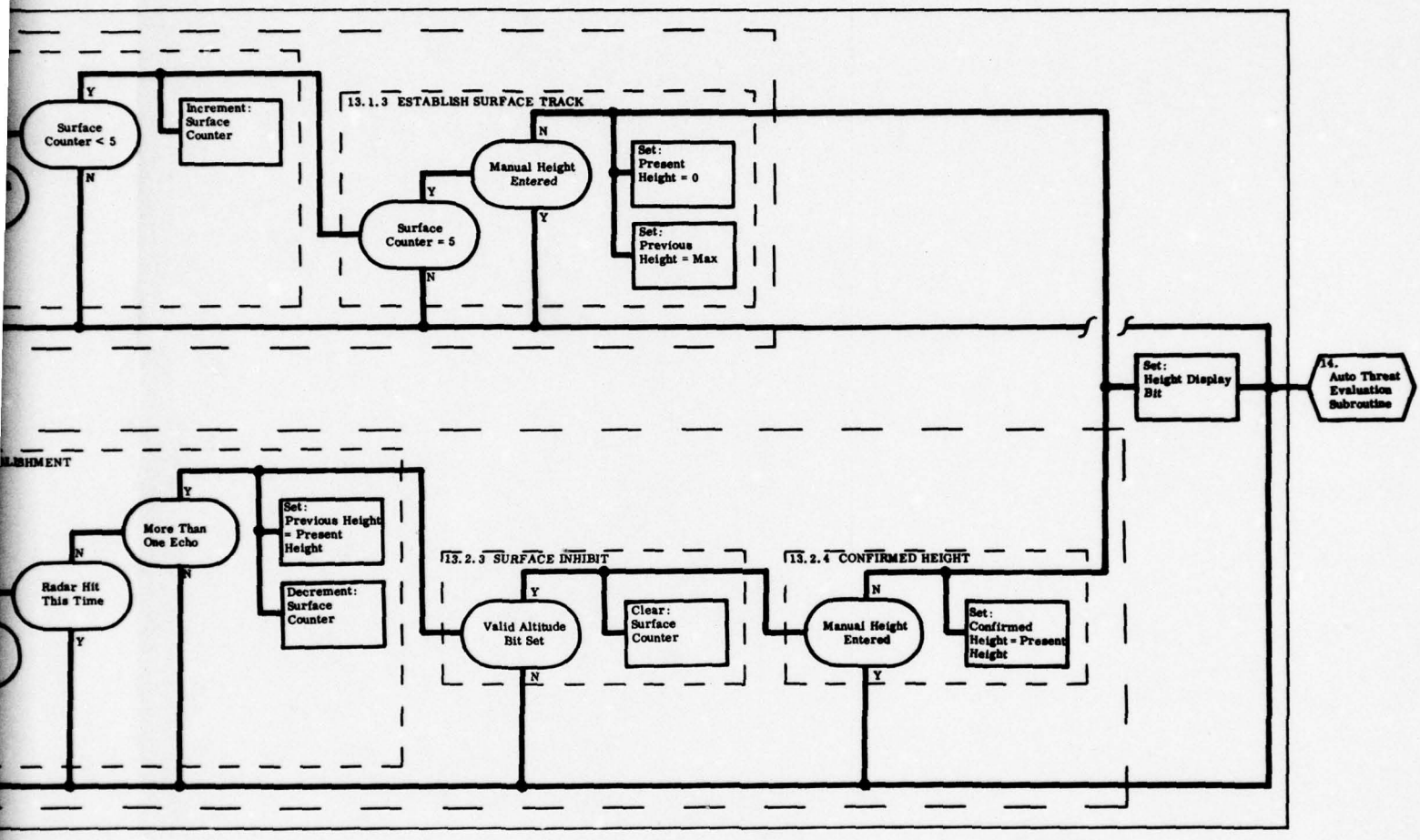


FIGURE 1
PROGRAM FLOW DIAGRAM FOR HEIGHT PROCESSING SUBROUTINE

3.1.5.3 Major Routine Program Flow Diagrams and Functional Flow Chart. These Major Routine Program Flow Diagrams and Functional Flow Charts shall contain one block for each subroutine which will be illustrated by a separate diagram. They shall also contain decision symbols, where applicable, to illustrate decision criteria for entry into the subroutine. Flow lines shall be used to illustrate flow between the subroutines. Where direction of flow is not readily apparent, arrowheads shall be used to indicate direction of flow.

3.1.5.4 Subroutine Program Flow Diagrams and Functional Flow Chart. The subroutine Program Flow Diagrams and Functional Flow Charts shall illustrate, by appropriate programming symbol, each individual decision, each individual action to be performed in the process illustrated by the diagrams, and the tie-ins to the sub-functional diagram(s) that produced the input and received the output.

3.1.5.5 Numbering Technique. Each grouping block at each level shall have an identifying number. Decimal indenture shall be used to indicate subordination of groupings. New routines shall be identified by name only. Numbers between the first and second decimal shall indicate the major subroutines within the major routines. Numbers between the second and third decimal shall indicate subroutines within the major subroutines. Additional subordinate numbering shall be used as required to identify the subordinate groupings within the subroutines. An example of the numbering technique is provided in Figure 1.

3.1.6 Standard Symbols

3.1.6.1 The standard symbols in MIL-STD-682 shall be used in the preparation of flow charts (machine generated diagrams excepted). These symbols shall not be redefined, nor other symbols substituted for the defined functions. If additional symbols for functions not defined are required for a specific application, such additional symbols may be used, provided the procuring agency has approved their use.

3.1.7 Symbol Template

3.1.7.1 All of the standard symbols can be drawn with the aid of the standard symbol template specified in MIL-STD-682. The template should be used in the preparation of all flow charts. Symbols available on the standard template are illustrated by Figure 3.

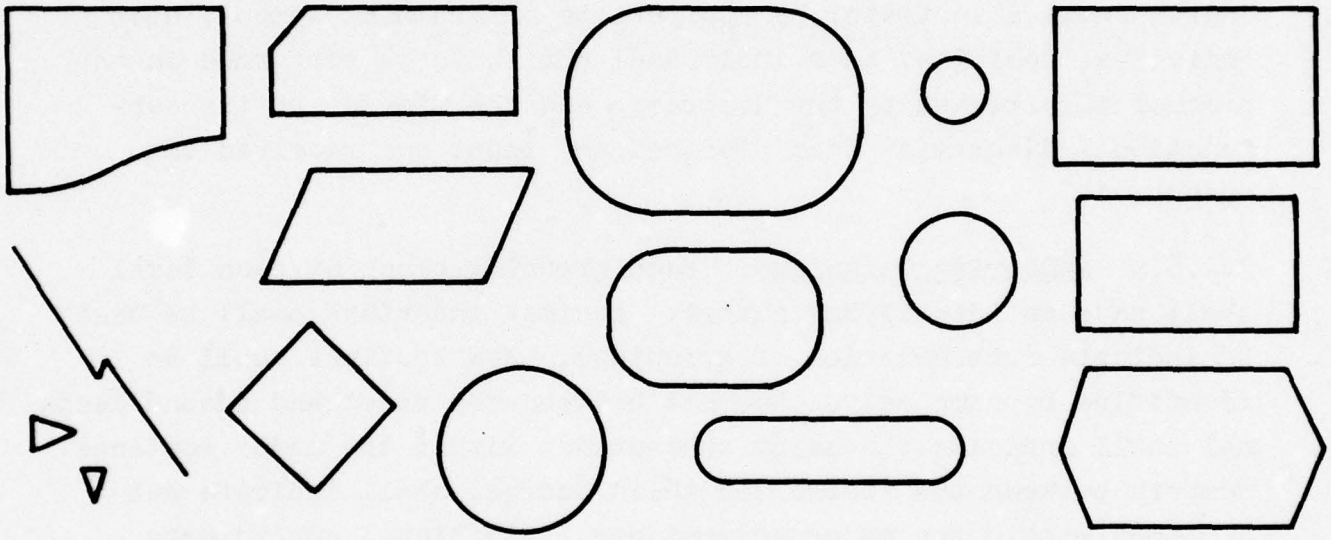


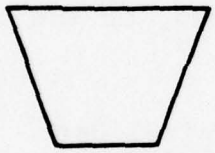
FIGURE 3
TEMPLATE FOR FLOW CHART SYMBOLS

3.1.8 Program Flow Diagram and Functional Flow Chart Symbols

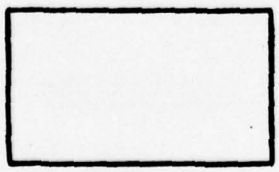
3.1.8.1 Symbols are used on Program Flow Diagrams and Functional Flow Charts to represent the functions of a data processing program or system. Basic symbols are established for the basic functions that are ordinarily included in high-level flow charts; i.e., input/output, processing, and annotation. Specialized symbols are established for the detailed functions that are ordinarily included in low-level (subroutine) flow charts.

3.1.8.2 Basic Symbols. The basic symbols are as follows:

(a) Input/Output Symbol. The symbol shown below represents the basic input/output function (I/O); i.e., the making available of data for processing (input), or the recording or displaying of processed data (output).

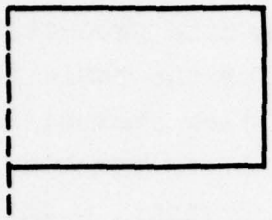


(b) Processing Symbol. The symbol shown below represents the basic processing function; i.e., the process of executing a defined operation resulting in a change in value, form, or location of data.



(c) Annotation Symbol. The symbol shown below represents the basic annotation function; i.e., the addition of descriptive comments or explanatory notes as clarification. The dotted line may be drawn either on the left and down (as shown), on the right and down, on the top and left, or on the top and right. It is

connected to the flowline, at a point where the annotation is meaningful, by extending the broken line in whatever manner is appropriate.

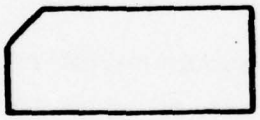


3.1.8.3 Specialized Symbols. The specialized symbols are as follows:

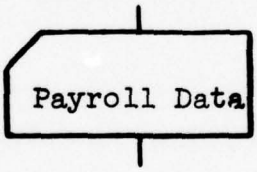
(a) Input/Output Symbols. The specialized input/output symbols represent the I/O function and, in addition, denote the medium on which the data is recorded or displayed or manner of handling data, or both. The input/output symbols are:

(1) Data Card Symbol. The symbol below represents an I/O function in which the medium is punch cards, or the like. The symbol also represents a card file.

Symbol

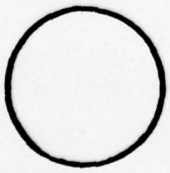


Typical Use



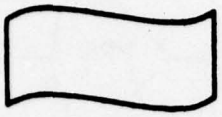
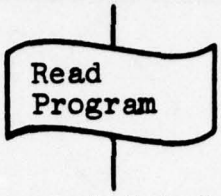
(2) Magnetic Tape Symbol. The symbol shown below represents an I/O function in which the medium is magnetic tape.

Symbol

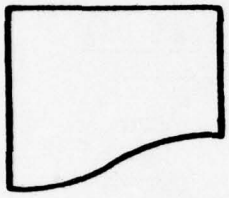
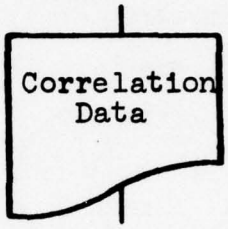


Typical Use

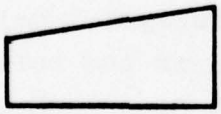
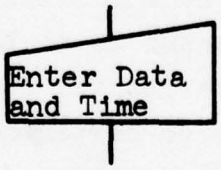
(3) Punch Tape Symbol. The symbol shown below represents an I/O function in which the medium is punch tape.

Punch Tape Symbol	Typical Use
	

(4) Printed Document Symbol. The symbol below represents an I/O function in which the medium is a machine printed document.

Symbol	Typical Use
	

(5) Manual Input Symbol. The symbol shown below represents an I/O function in which the data is entered manually at the time of processing, by means of on-line keyboards, switch settings, push buttons, etc.

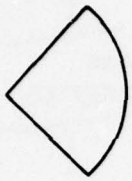
Portion of Symbol	Typical Use
	

NOTE

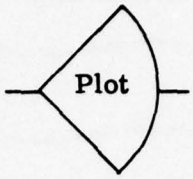
This symbol can be drawn with the template, but it is not defined as a Military Standard symbol.

(6) Display Symbol. The symbol shown below represents an I/O function in which the data is displayed for human use at the time of processing, by means of on-line printers, console printers, video devices, plotters, etc.

Portion of Symbols

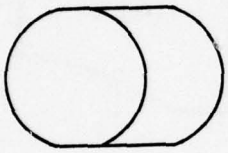


Typical Use

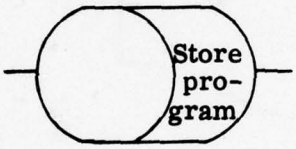


(7) Mass Storage Symbol. The symbol shown below represents an I/O function utilizing auxiliary mass storage of data that can be accessed on-line, e.g., magnetic drums, magnetic discs, or other computer.

Symbol



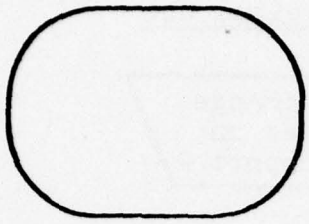
Typical Use



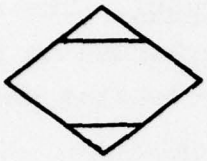
(b) Processing Symbols. The specialized processing symbols represent the processing function, and in addition identify the specific type of operation to be performed on the data. The processing symbols are:

(1) Decision Symbol. The symbol shown below represents a decision type operation that determines which of two or more alternate paths is to be followed. This symbol may contain, upon option, a horizontal subdivision line near the bottom, below which may be entered pertinent or informative data such as the data unit on which the decision is made, an input or output condition, or a sense switch setting.

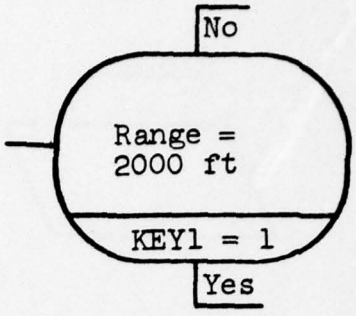
Symbol

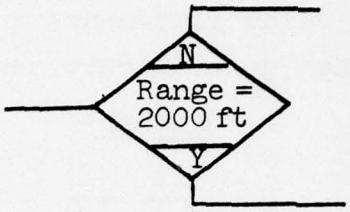


or



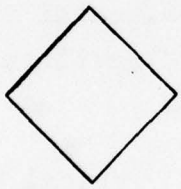
Normal Decision Use



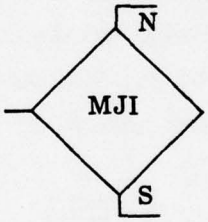


(2) Switch and Branch Symbol. The symbol shown below represents a switching type of operation, e.g., the routing of input data to either of two output paths depending upon the setting of the switch, either "set" (S) or "normal" (N).

Symbol

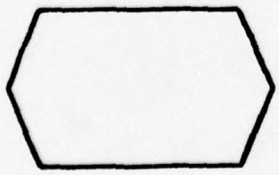


Typical Use

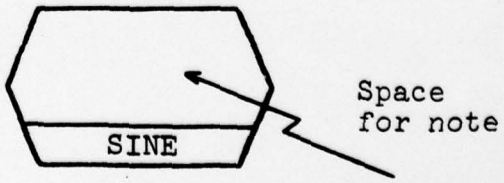


(3) Subroutine Symbol. The symbol below represents a reference to a subroutine. This symbol may contain, upon option, a horizontal division line near the bottom, below which pertinent or informative data may be entered; e.g., the subroutine label.

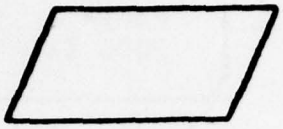
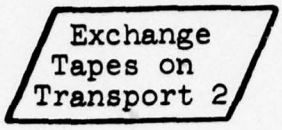
Symbol



Typical Use



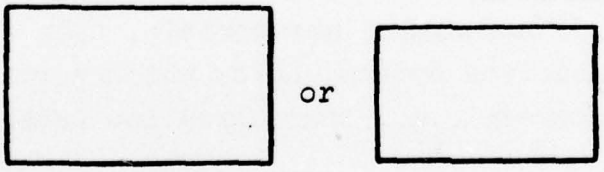
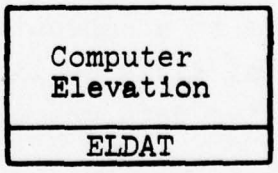
(4) Manual Operation Symbol. The symbol shown below represents any off-line process geared to the speed of a human being.

Symbol	Typical Use
	

(5) Auxiliary Equipment Operation Symbol. The symbol shown below represents an off-line operation performed on equipment not under direct control of the central processing unit.

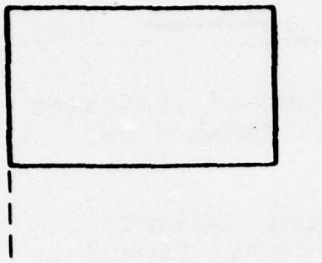
Symbol	Typical Use
	

(6) Non-Auxiliary Equipment Operation Symbol. The symbol shown below represents an on-line operation performed on equipment under direct control of the central processing unit. This symbol may contain, upon option, a horizontal subdivision line near the bottom, below which may be entered pertinent or informative data, such as names of data being processed.

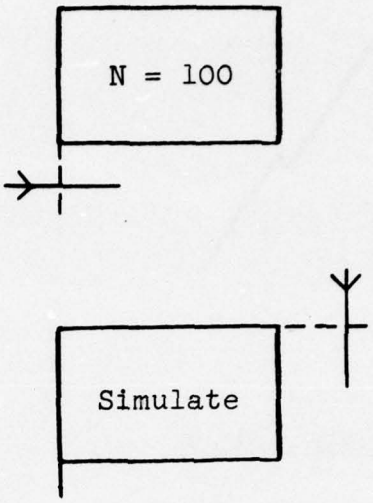
Symbol	Typical Use
	

3.1.8.4 Assertion Symbol. The symbol shown below indicates a special condition existing at a certain point along a flow line.

Symbol



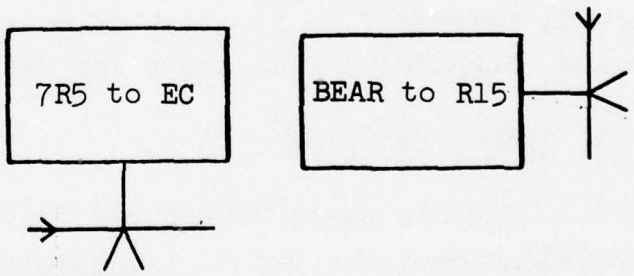
Typical Uses



3.1.8.5 Insertion Symbol. The symbol shown below indicates the insertion of a processing symbol into a flow line; the single line drawn to the symbol represents both the input line and the output line.

Symbol

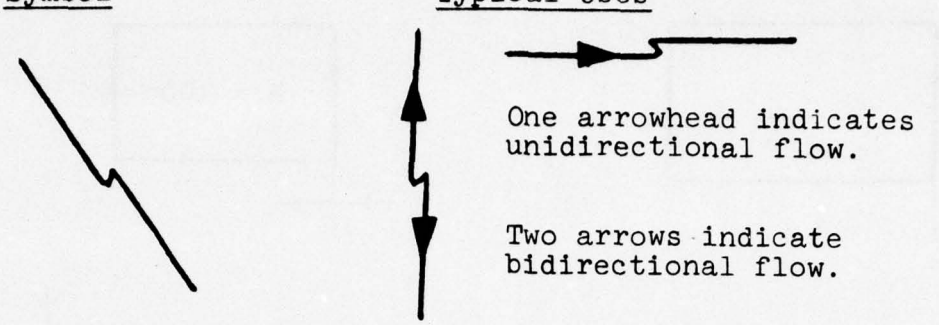
Typical Uses



NOTE

This symbol is intended for use in updating flow charts and should not appear in any final flow chart.

3.1.8.6 Communication Link Symbol. The symbol shown below represents an I/O function in which data is transmitted automatically from one location to another. The symbol shall always be drawn with arrowheads to denote the directions of data flow.



3.1.9 Symbol Sizes

3.1.9.1 When reduced 50 percent, the size of each symbol shall be as illustrated on the sample block diagrams. The sizes illustrated on the template are considered appropriate for the "up" or non-reduced size preparation of any flow chart.

3.1.10 Symbol Orientation

3.1.10.1 All of the symbols shall be oriented as illustrated in Figure 1, Program Flow Diagram for Height Processing Routine, and Figure 2, Functional Flow Chart for Height Processing Routine.

3.1.11 Symbol Grouping

3.1.11.1 Symbols shall be grouped functionally to illustrate subfunctional actions and to illustrate the relationships between the subfunctional actions. Groupings shall be selected to have a single input and two outputs as illustrated in Figure 1. Two basic groupings are required in the subroutine flow diagram to illustrate the subroutine and to facilitate the preparation of and reference to the functional descriptions. These are:

(a) Grouping of decision groupings and associated action groupings that form an identifiable sub-subroutine.

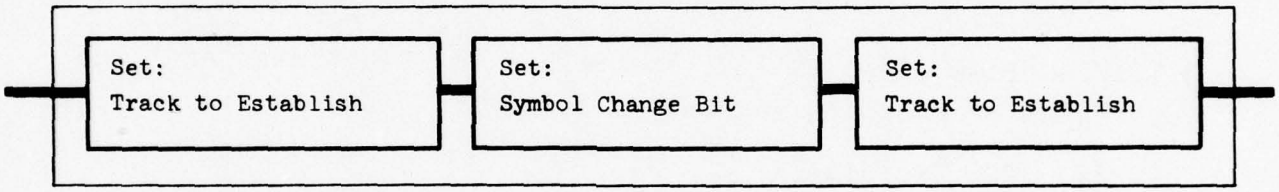
(b) Grouping the identifiable sub-subroutines into higher order subroutines.

Individual decision symbols which control the entry into one or another sub-subroutine need not be contained in a box.

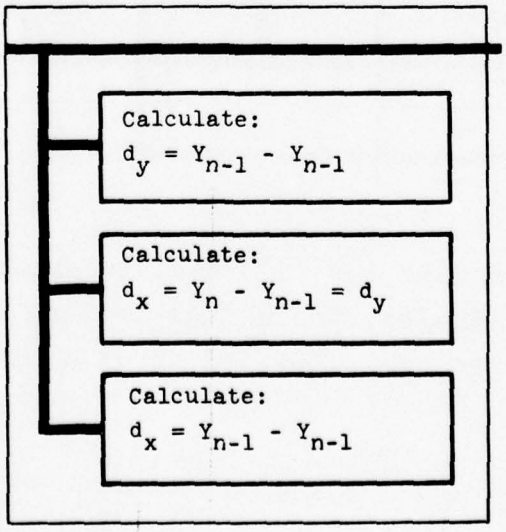
3.1.12 Flow Lines

3.1.12.1 Flow lines on the Program Flow Diagram will be controlled to illustrate the sequence to be followed by the programmer during the coding process. To facilitate the coding process, both the fixed sequences and the optional sequences must be illustrated clearly on the Program Flow Diagram as follows:

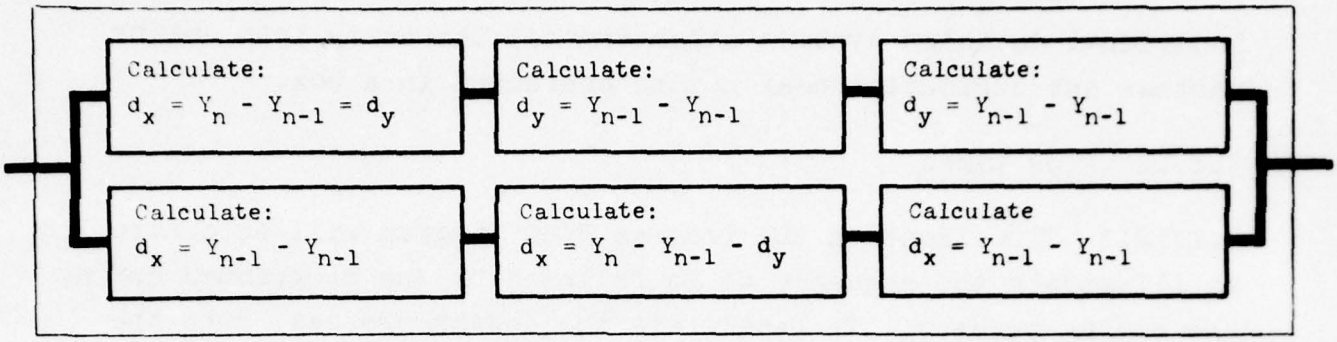
(a) Actions which must be performed sequentially:



(b) Actions which must be performed at this time in the program but can be performed in any sequence. (The person doing the coding may find an opportunity to improve program efficiency by changing the sequence of action.)



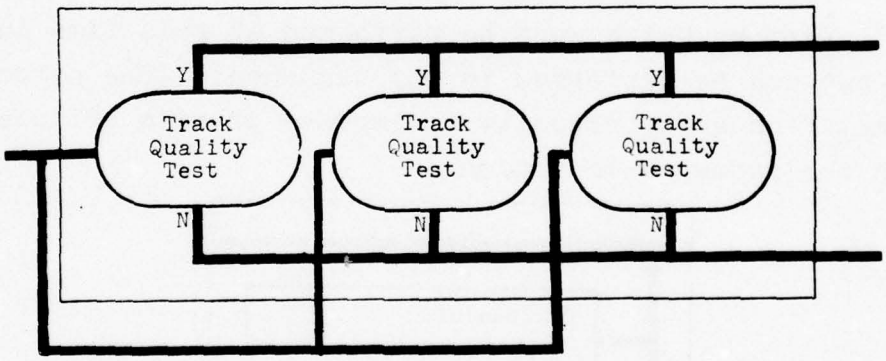
(c) Sequences of actions which may be performed in optional sequence:



On Functional Flow Charts option sequence is prohibited, since these charts illustrate the order in which the options were exercised.

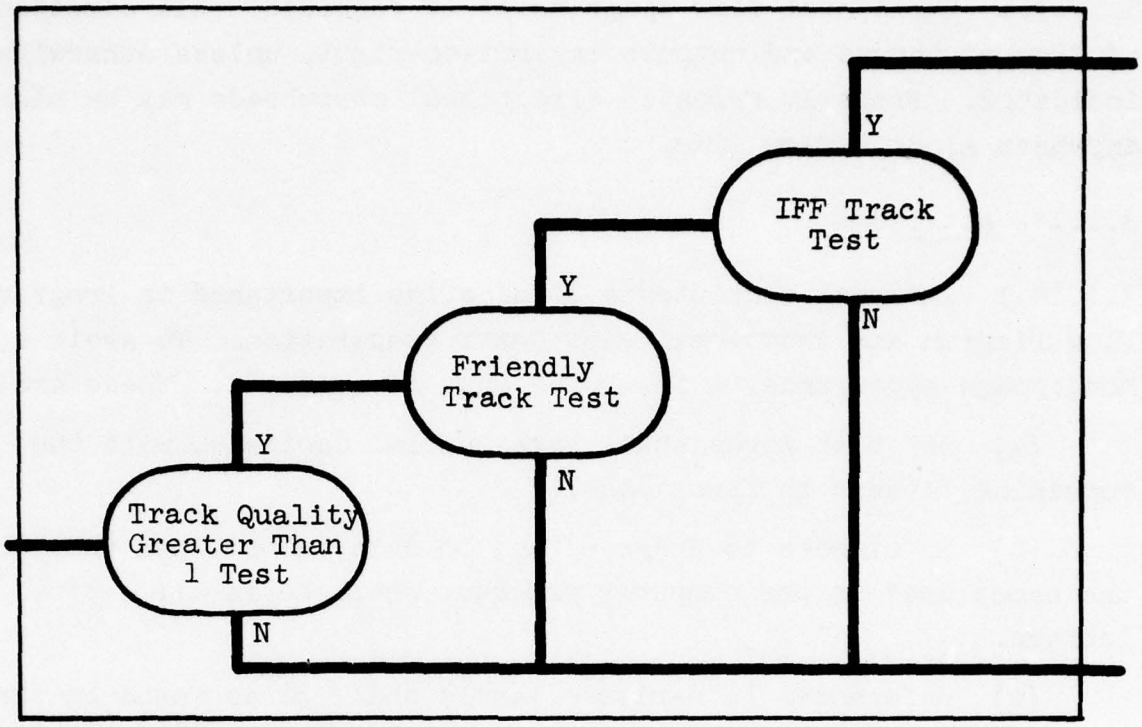
3.1.12.2 Flow lines will be controlled to illustrate "and" and "or" types of decision actions as follows:

(a) "Or Decision":



On Program Flow Diagrams the illustration implies optional sequence, On Functional Flow Charts, the actual testing sequence is left to right, with the first test provided in left symbol.

(b) "And" Decision:



3.1.12.3 Straight lines, vertical or horizontal, shall be used to show the flow of control or of data between the symbols on a flow diagram. Symbols requiring two output lines shall normally have both output lines perpendicular to the input line. None of the lines--input or output--are considered to be inherently the part of any symbol.

3.1.12.4 Connection of the lines to the grouping boxes are indicative of program sequence. Lines passing left to right through a grouping box indicate that the decision or action listed in the box must be performed before the program can proceed. Lines entering a grouping box from the top, or lines entering a grouping box at the left, with no line departing the right of the box, indicate an action that must be performed at this time, but that subsequent program operations within this subroutine are not dependent on the results of the action.

3.1.13 Direction of Flow

3.1.13.1 Horizontal flow diagramming is required. All direction of flow of inputs and outputs is left-to-right, unless otherwise indicated. Properly oriented directional arrowheads may be placed anywhere along a flow line.

3.1.14 Notations

3.1.14.1 Internal consistency is of prime importance in Program Flow Diagram and Functional Flow Chart preparation. To avoid a hodgepodge appearance, a few rules must be observed. These are:

(a) All text words shall have initial capitals, with the remaining letters in lower case.

(b) References to programs and to data units, when they are the names used in the computer program, shall be in all capital letters.

(c) References to hardware labels shall be as found on the hardware.

(d) No question mark shall be placed at the end of the text in a decision box; the symbol itself indicates a question.

(e) Text shall be condensed so as to fit within the symbols, but abbreviations avoided, if possible.

(f) Minimize mathematical notation unless expressing complete equations. Where possible, text shall be in ordinary English in terms that can be easily understood.

(g) Questions shall be phrased (in decision boxes) so that they can be answered with a "Yes" or "No". Other responses are permissible; e.g., , , =, and combinations thereof. If the decisions are expressed in English words, the first letter of each word shall be capitalized.

(h) Action illustrated by an action box shall be indicated by a statement of the action in the upper left corner of the box; i.e., Set:, Subtract:, Compute:, Increment:, etc. The action to be performed shall be indicated on the following lines within the box.

3.1.15 Subroutine Grouping

3.1.15.1 Major routines within the program may have single input and single output, or single input and two outputs. Major routines shall be subgrouped as illustrated in Figure 4 to identify the subroutines within the major routine, and to illustrate the relationship and flow between the subroutines.

3.1.16 Data Management Control Block

3.1.16.1 Each diagram shall have a data management control block. The basic identifying number (first eight digits) will be assigned by FCPC. The drawing control number (next four digits) are assigned by the contractor and recorded with TDS Computer Program Manager's Documentation Library. The first two digits of the document control number shall indicate the major routine (01 through 99); the second two digits shall indicate the subroutine (01 through 99) within the routine.

3.1.16.2 The fifth digit identifies the type of document as follows:

1 – Operational Specification
2 – Program Flow Diagram
3 – Functional Description of Logic Flow
4 – Functional Flow Chart
5 – Program Routine
6 – Test Routine
7 – Program Listing
8 – Assembly Listing
9 – Simulation Program

3.1.16.3 The interchangeability designator (last two digits) indicates the degree of interchangeability of the various configurations of the routine or subroutine as follows:

-X01 through -X09 enables identification of the addition of a feature which is interchangeable in a program, with the addition or loss of the feature as the only consequence of interchange;

-X10 through -X90 (increments of 10) indicates non-interchangeable items;

-X10 through -X19, etc. (increments of 1) indicates interchangeable items within the non-interchangeable items.

3.1.16.4 The revision letter following the number indicates a change that does not affect interchangeability nor add or delete a feature; i.e., error correction or improved approach.

3.1.17 Sizes

3.1.17.1 Each flow diagram shall be prepared on a 22x56 sheet (maximum), with the type or lettering size suitable for 2:1 reduction maintaining 0.060 type at the reduced size.

3.1.18 Layout

3.1.18.1 The layout for a flow diagram is illustrated in Figure 1.

3.2 FUNCTIONAL DESCRIPTION

3.2.1 This subsection specifies the requirements for preparation of Functional Descriptions of Logic Flow. The Functional Description is a documentary of the logic and logic flow illustrated by the Program Flow Diagram. Additionally, it describes the requirement in the Operational Specification that is satisfied by the portion of the Flow Diagram being described.

3.2.2 Preparation

3.2.2.1 One statement shall be prepared for each numbered block on each diagram. The statement shall be a brief description of decisions or actions being performed by the numbered functional grouping being described.

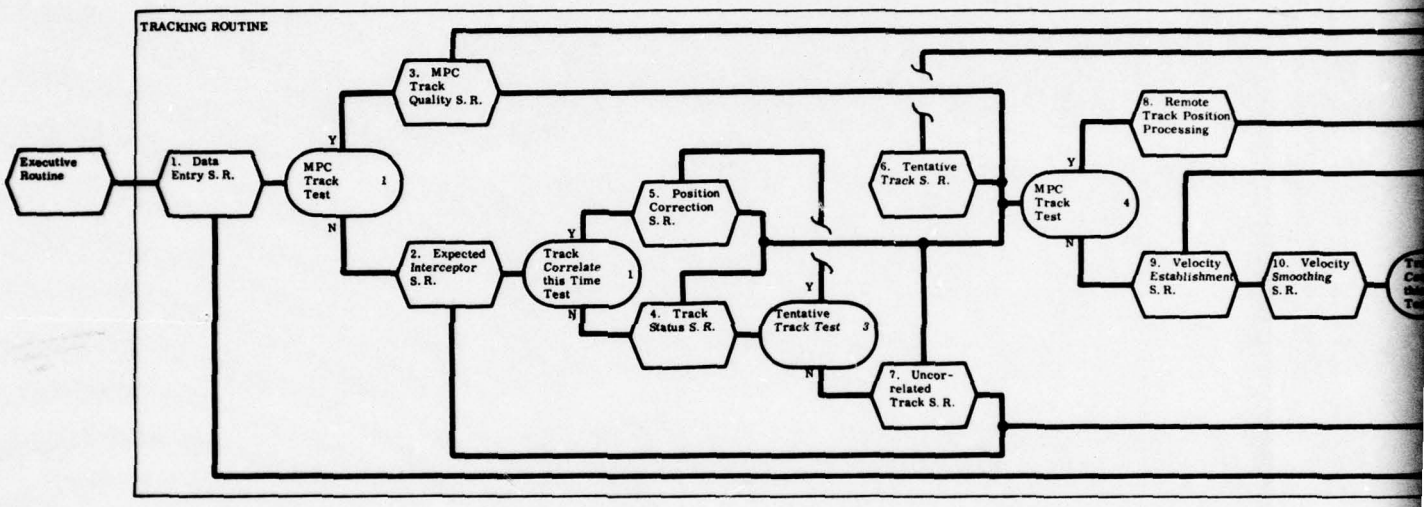
3.2.2.2 For higher order functional groupings or higher order diagrams (those containing subordinate functional groupings), one statement describing the interrelations of the subfunctional groupings contained within the higher order grouping shall be provided.

3.2.2.3 Statements, which may range from single sentences to paragraphs, as required, shall be brief descriptions of the technical details. Lengthy descriptions are neither required or desired.

3.2.3 Text

3.2.3.1 Text shall be concise. Verbiage is neither desirable or permissible. It shall provide the "what it does", and "when it does", and "how it does" description of each block.

3.2.3.2 The text shall be detailed sufficiently to permit the reconstruction of the Program Flow Diagram from the Functional Description. The sample portion of a Functional Specification corresponds directly to the Program Flow Diagram example (Figure 1).


TRACKING ROUTINE
Executive Routine
1. Data Entry S. R.
MPC Track Test 1
Y
N
3. MPC Track Quality S. R.
2. Expected Interceptor S. R.
Track Correlate this Time Test 1
Y
N
5. Position Correction S. R.
4. Track Status S. R.
Tentative Track Test 3
Y
N
7. Uncor-related Track S. R.
6. Tentative Track S. R.
MPC Track Test 4
Y
N
8. Remote Track Position Processing
9. Velocity Establishment S. R.
10. Velocity Smoothing S. R.

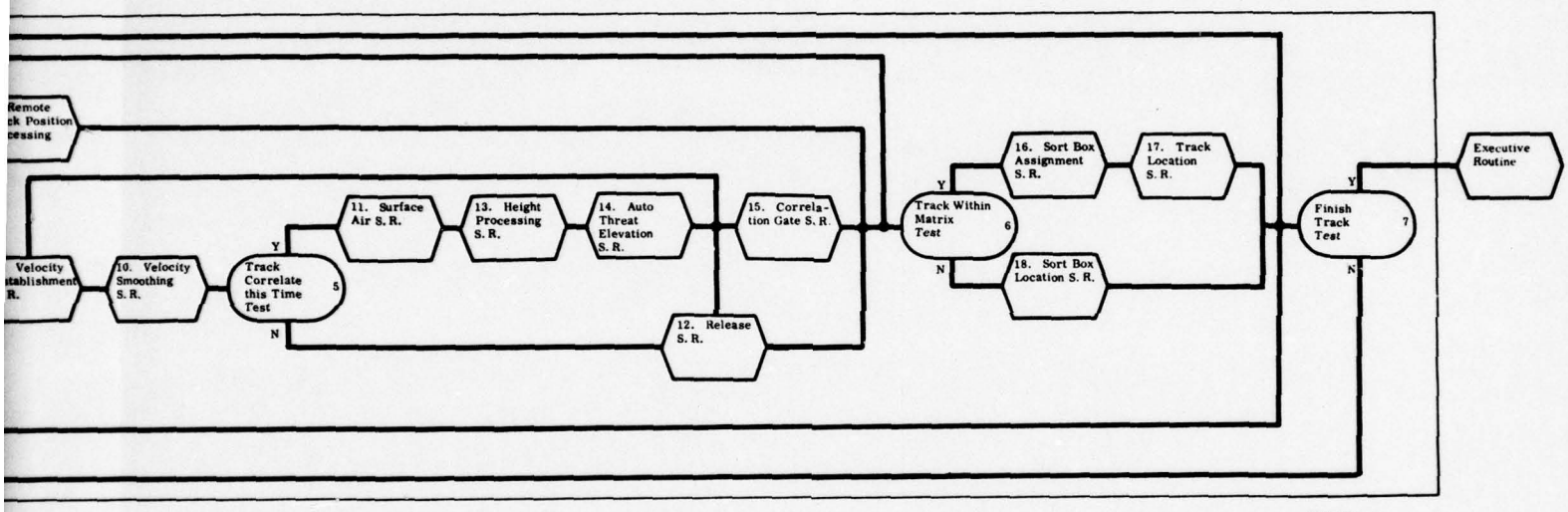


FIGURE 4
TRACKING ROUTINE PROGRAM FLOW DIAGRAM

3.2.4 Subdivision

3.2.4.1 The Functional Specification shall be divided into chapters, sections, and paragraphs which are numbered to correspond with the sub-functionalization numbering of the Program Flow Block Diagram. Separate chapters shall be provided for each of the major routines that make up the complete program; i.e., navigation, tracking, message processing, etc. Overall description of the complete program, including a description of the interrelations between the major routines, shall be contained in Chapter 0. Each chapter shall be divided into sections, with one section provided for each of the major subroutines that make up the major routine. For example, the tracking routine chapter would contain a section for each of the subroutines concerned with velocity establishment, velocity smoothing, remote track processing, height processing, etc. Overall description of the major routine, including a description of the interrelations of the subroutines, shall be combined in Section 0.

3.2.4.2 The chapter and section or division and numbering technique specified provides rapid correlation of data between the Functional Description and the Program Flow Diagram, and must be maintained to continue this correlation.

3.2.4.3 Each chapter and section shall begin on a new page. (It is realized that some sections will not require a full page. This is acceptable, and no attempt should be made to add verbiage simply to fill a page.)

3.2.5 Rules for Grammar

3.2.5.1 Technical accuracy and adequacy are the prime requisites of the functional description. The following rules for grammar are provided in the interest of consistency.

(a) Third person.
(b) Present tense (use "when" prepositional phrase for time).
(c) Active voice.
(d) Indicative mood.

3.2.6 Summary

3.2.6.1 By following the few simple rules provided in the preceding paragraphs all levels of text will be covered.

3.2.6.2 The text describing the overall program will define the major routines involved in the program, and describe their interrelationship.

3.2.6.3 The text describing each major routine will define the major subroutines involved, describe the logic that determines when the subroutines shall be employed, and further, describe the function of each.

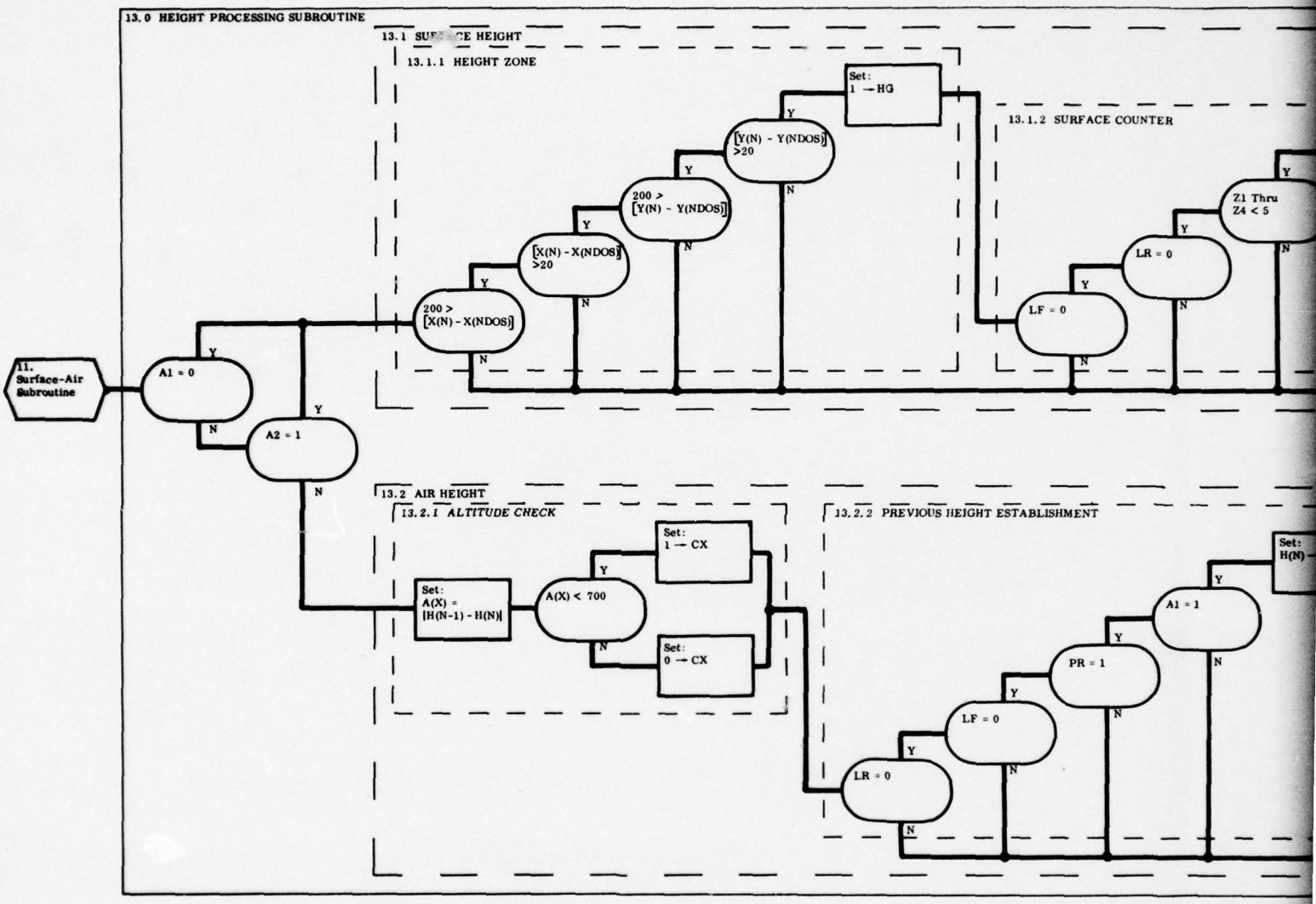
3.2.6.4 The text describing each subroutine will define the sub-subroutines, the logic that determines when the sub-subroutine shall be employed, the logic used in each of the decision functional groupings, and the actions to be performed by these functional groupings.

3.2.6.5 Each statement shall provide the answers to the following questions: (1) what it does, (2) when it does it, and (3) why it does it.

3.2.7 Sample Functional Description

3.2.7.1 The following example is provided as a guideline for the preparation of functional descriptions of the various levels of functional groupings. Note the requirement for concise narrative description rather than verbiage prose.

3.2.7.2 The paragraphs are numbered to correspond to the blocks on Figure 1 and the text describes the decisions and actions that occur.


13.0 HEIGHT PROCESSING SUBROUTINE
13.1 SURFACE HEIGHT
13.1.1 HEIGHT ZONE
13.1.2 SURFACE COUNTER
11. Surface-Air Subroutine
A1 = 0
A2 = 1
200 > [X(N) - X(NDOS)]
[X(N) - X(NDOS)] >20
200 > [Y(N) - Y(NDOS)]
[Y(N) - Y(NDOS)] >20
Set: 1 → HG
LF = 0
LR = 0
Z1 Thru Z4 < 5
13.2 AIR HEIGHT
13.2.1 ALTITUDE CHECK
13.2.2 PREVIOUS HEIGHT ESTABLISHMENT
Set: A(X) = |H(N-1) - H(N)|
A(X) < 700
Set: 1 → CX
Set: 0 → CX
LR = 0
LF = 0
PR = 1
A1 = 1
Set: H(N)
Y
N

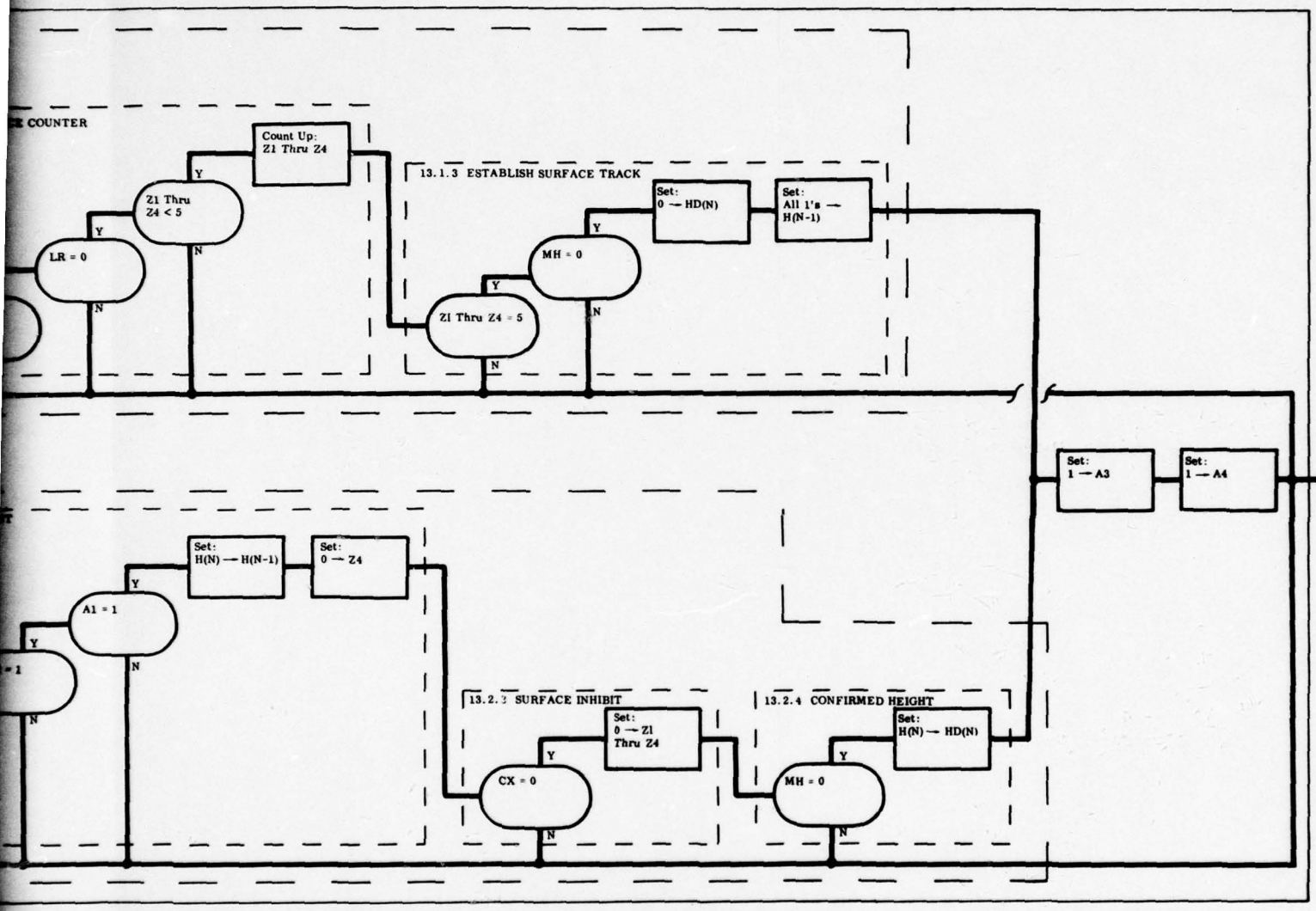


FIGURE 2. FUNCTIONAL FLOW CHART FOR HEIGHT PROCESSING ROUT

13. HEIGHT PROCESSING

The Computer-detector performs the height finding process by measuring the time-difference between the direct path and the multipath return echo. Results of the CD test is stored in Height Accumulator. In addition to the multipath target feature, other criteria are established for Height Processing. Conditions are provided in 13.2 and 13.3.

13.1 Surface Height

13.1.1 Confirmed surface targets (0 height) are processed in this grouping. Requirements for confirmed surface targets are as follows:

(a) Target within height finding zone.
(b) Target not locked with front tag.
(c) Target not locked with rear tag.
(d) No manual height entered.
(e) Valid one echo report.
(f) Surface counter indicates 5 valid single echo returns.

13.1.2 In this section the target is tested for position within the height finding zone. The height finding zone is defined as further than 20 miles, but closer than 200 miles. When the criteria is met, the target in Height Finding Zone Bit is set.

13.1.3 Rules for counting valid echo for surface counter are:

(a) Ignore targets in obscuration zone (front or rear tag set).
(b) Increment valid echos.
(c) Ignore garbled echos.
(d) Reset to 0 for valid height (multiple echo).

In this grouping the front and rear tag are tested for target location outside obscuration zone, the surface center is tested to verify that the surface count is less than 5, then the surface counter is incremented by one. Subsequently, the surface counter is tested for being equal to five.

13.1.4 In this section the surface counter is tested for adequate correlation of valid surface echos and, provided that manual height has not been entered, establishes the target as a surface target by setting present height to 0 and previous height to maximum. This ensures a difference between present height and previous height greater than 700 feet thus preventing false returns from establishing this track as an air track.

3.3 PROGRAMMING PRACTICES

3.3.1 Rules for good programming are helpful for small programs coded by a single programmer; they are essential for large programs coded by many programmers. The practices described in the following paragraphs are designed to unify large programs coded by many programmers. Any exceptions to these rules must be approved by a programming coordinator and distributed to all the programmers. Inter-program coding practices are sufficiently well documented in "Naval Tactical Data System", Volume 2, CS-1 Manual, Appendix B, and will not be listed here.

3.3.2 A variable, assigned a name by the programmer, can equal numeric data or alphanumeric labeling information. Variables may assume a succession of values during execution of the program.

3.3.2.1 Naming Variables. The data names of all variables shall be symbolic. Variables common to more than one major routine must have the same symbolic name in all the routines in which they appear. Also, all data names shared by several routines should begin with the same two letters. The remaining characters must be mnemonic. Data names which appear in only one major routine shall all begin with two letters uniquely assigned to that routine. For example, a variable used only in the executive module might be named EXRTF, where EX stands for Executive module and RTF is Real Time Flag. Similarly TKRTD is the Real Time Decode from the Tracking Routine. All variables used only by the Executive Routine should begin with EX, and all those used only by the Tracking Routine should begin with TK. Appendix R of the CS-1 Programmers Guide, contains a "Restricted Word List"; data names should not be any of the words on this list.

3.3.2.2 Definition of Variables Assigned as Data Units. All data units common to a routine shall be defined at the beginning of the source program. Data names defining tables shall be grouped together, as shall data names defining variables, indices, titles, and switches.

3.3.2.2.1 In coding programs which require a great deal of core, some programmers may try to reduce core use by sharing scratch storage between routines. This is accomplished by transferring a scratch storage data unit to a second program as an input parameter so the second program can share the first program's scratch storage. This practice can be a source of many errors and should be avoided whenever possible.

3.3.3 Operations define the action the program is to take for one instruction.

3.3.3.1 Operation Procedures. It is necessary to provide rules to insure uniform coding by different programmers. Since programmers will have different coding styles (and specialties), more rigid regimentation of coding practices will be necessary in this section than at any other level. Further, uniform coding will help insure that an optimal method of coding is employed, since coding which is used repeatedly should encourage a higher level of effort by the programmer. The resulting coding can be of two types:

3.3.3.2 Brief Groups of Coding. Some judgment must be exercised before deciding that a given collection of instructions is used often enough to be made into a procedure. It may save machine-run time to simply repeat the collection of instructions at each use, rather than to call a procedure which performs the functions of the instructions. Coding of a collection of instructions should be reviewed by a systems analyst for optimal coding. Its subsequent use throughout all the routines must be as uniform as possible. Nonstandard use of poly operations in such coding should be avoided, since mono operations will provide more efficient coding in these cases. These collections of instructions should be given names or be labeled (by comment cards) so they may be referenced.

3.3.3.3 Procedures. Those groups of instructions which are repeated frequently and which are long enough to justify the time consumed by linking instructions should be written in procedure form. No procedure shall change the value of a register or variable unless this is its specific function. Thus, any intermediate

storage needed for a procedure must be specifically assigned. The arbitrary unique names assigned to input, output, or exit parameters in defining a procedure should be mnemonic. If the number of input, output and exit parameters exceeds the limit of the compiling system in use, the usual technique to bypass this restriction is to pack several variables into a newly defined data unit and then use the new data unit as a parameter to the procedure. This should be avoided if possible, but if it is the only solution to the problems caused by this restriction, then the newly defined data unit must not be used for any other purpose and must be documented by comment cards (see 3.4). All procedures must be documented by comment cards for subfunctional routines as defined in 3.4, where they apply to the procedure. Finally, abnormal exits should be grouped at the end of the procedure and should not interrupt the normal flow of the procedure.

3.3.4 Diagnostic Output

3.3.4.1 Errors during execution can be of two kinds:

(a) Machine errors — Since machine errors rarely occur on equipment which is tested daily, the program need only test for human error.

(b) Human error — Input to the programs will be provided in part by operators of the equipment. The programs will test that all such input is within the range of acceptable data. When incorrect data is found, the program must output a diagnostic message. Incorrect input and the errors it produces are called error conditions.

3.3.4.2 Those error conditions which are common to more than one routine shall have a uniform diagnostic message which gives the error, the state of the computer registers (if appropriate), the absolute location where the error occurred, and the name of the routine in which the error occurred. In order to avoid sharing duplicate output statements in several routines, an error procedure shall be supplied. Such a procedure would be called by the

routines when an error condition is encountered. One of the arguments used to enter the error procedure would indicate the type of error encountered.

3.3.4.3 Those errors not common to several routines should follow the general print format of the error procedure, but they may be a part of the routine where they occur, rather than being a part of the error procedure. They should be placed at the end of the routine so they will not interrupt the flow of the program for a normal run.

3.3.5 Linking, Branching, and Overwriting

3.3.5.1 The contents of the index registers, location counter, arithmetic registers, sense switches, and overflow, underflow, and divide check sensors all reflect the state of the computer at execution time. When linking procedures, some of these must remain unchanged. Uniform coding practices demand that the programmers agree on a list of those registers and counters which must be protected within a procedure. Then the contents of the registers and counters which are to be protected are saved at the entry point to the procedure and are restored at the exit points of the procedure.

3.3.5.2 A procedure may not change an instruction in a routine. If it is necessary to overwrite a particular instruction word in another routine at execution time, the flow of the program must transfer to the routine to be overwritten. Thus, the instructions causing the overwriting will be executed in the routine to be overwritten. Even so, all such overwriting must be properly documented by Comment Cards and should be avoided whenever possible.

3.3.5.3 The main flow of the program should proceed down the page (of the listing) or sequentially through core storage. Major branches will be at the end of the program. In general, the flow of the program should reflect as closely as possible the direction of the flow chart.

3.3.5.4 Instructions to which the program flow is transferred must have a label. This label should be assigned a symbolic name;

absolute locations should be avoided. In particular, no subprogram should refer to a branch point as "N operations from this one" as in FAP we might write "TRA A + 4", meaning transfer to four locations after symbolic location A.

3.3.5.5 The symbolic labels for transfer points can be a debugging aid. The symbolic label should reflect the particular step in the program where the operation is performed. It would be particularly helpful if the label referred to the numbering system of the Flow Charts.

3.4 COMMENT CARDS AND FLOW CHART CARDS

3.4.1 Definition

3.4.1.1 Comment Cards. The CS-1 Compiler allows the programmer to insert coding explanations in the body of a program by an operator called "Comment." This comment is printed in the listing in the same position with respect to the coding as it had in the source deck.

3.4.1.2 Flow Chart Cards. The CS-1 Compiler allows the programmer to insert FC Cards in the body of the program. At compile time, these cards are printed with the listing as were the Comment cards, and they also can be printed as a special output from the listing in the form of a flow chart. In addition, Flow Chart Comment Cards ("FC Comment") will cause the compiler to print the FC Comment information in the body of the listing and also print the information to the extreme right of the page containing the flow charts for that section of coding.

3.4.2 Use

3.4.2.1 Comment and Flow Chart Cards are inserted within a source program deck as an aid in all aspects of maintenance, reliability, and revisability of the source program. Since the comments on the cards are an integral part of the source program listing, they are the form of documentation most readily available to the programmer. Revisions of the source program will require revisions of

associated Comment and Flow Chart Cards when a discrepancy exists between the changed program and the original Comment Cards.

3.4.2.2 Comment and Flow Chart Cards are to be prepared so that when printed by the compiler, there will be no midword break unless it is a syllable break at the end of the line, with the conventional hyphen (-) to indicate this. Also, comments should be in complete sentences when they are describing an operation or sequence of operations. Only labeling descriptions may be written in phrases.

3.4.3 Required Content

3.4.3.1 There are four kinds of Comment and Flow Chart Cards, catalogued by their position in the source program: (1) lead cards (2) subfunctional lead cards (3) internal comment cards, and (4) final comment cards. Each of the card types is described subsequently.

3.4.3.2 Lead Cards. Lead cards precede the program cards for each major routine of the program, and precede the cards for the complete program. The information required on the lead cards is identical for either use. Lead cards shall be of three types: (a) purpose definition cards (b) operating instruction cards, and (c) data unit description cards.

3.4.3.2.1 Purpose Definition Cards. Purpose definition cards provide the following information:

(a) A brief description of the function of the program (written as a Flow Chart Comment Card).

(b) A breakdown of the function of the program into subfunctions. This should be of the form: number, title, description of subfunction. "Number" must be the configuration control number and "title" must correspond to the title of that subfunction as listed in the functional specifications. The description may correspond to the functional specifications description, since it need serve only to distinguish one subfunction from another (written as a Comment Card).

3.4.3.2.2 Operating Instruction Cards. (Written as Comment Cards.) Operating instruction cards provide the following information:

(a) Input and output requirements. Since the details of these requirements appear in the documentation for the Testing Routine for this program, this description need not be detailed or referenced, but must be complete enough to supplement the purpose definition (see 3.4.3.2.1 (b)) in an understanding of the program. The source of the input and the destination of the output should be given.

(b) Auxiliary equipment needed. If special tapes, punched tape, cards, or cathode ray tube displays are needed by the program for input or output, the Comment Cards should list the equipment type and model number as well as give a unit designation when the executive system requires it.

(c) Sense switches or console switches which must be set at the beginning of the program or at intermediate points during execution of the program.

(d) List of operator action to interrupt the program because of error diagnostics. It is suggested that the Comment Cards list the type of error, the action to be taken, and the subfunctional program number (or numbers) that can give this error.

(e) Subroutines, identified by version numbers, which are not part of the subprogram breakdown or the system executive routine but which are used by the program and must be loaded with the program in order that the program will run.

(f) Core storage requirements for this routine and those subroutines in (e) above.

3.4.3.2.3 Data Unit Description Cards. All that data referenced by several subfunctional procedures throughout the program will be part of the system data design. Each table defined must be preceded by a Comment Card giving the functional description of the table in the program. Any nonstandard use of tables or their associated indices should be documented here. Single word variables should also be described here by Comment Cards.

3.4.3.3 Subfunctional Lead Cards. Subfunctional lead cards precede and describe the subroutines within the major routine illustrated by the individual Logic Flow Block Diagrams. Subfunctional lead cards shall be of three types: (a) purpose definition cards (b) operating instruction cards, and (c) data unit description cards.

3.4.3.3.1 Purpose Definition Cards. Purpose definition cards shall contain the following information.

(a) Description of the function of this subprogram in complete sentences (Flow Chart Comment Card).

(b) List of algorithms or techniques used to achieve the purpose of this subprogram. Error formulas on numeric alogrithms should be given (Comment Card).

(c) Details of input requirements with an example, if possible (Comment Card).

(d) List of the steps taken in this subprogram to test the input data and achieve the function of the subprogram. The step numbers must correspond to the sub-subfunctional breakdown appearing on the flow charts (Comment Card).

(e) Author of the original subprogram and responsible programmer of this revision (Comment Card).

(f) Filing number of the testing routine. Date of last test of this subprogram with its testing routine (Comment Card).

(g) List of the manuals, books, articles, etc., which may be of use in duplicating this function on another computer in another language. The Functional Specifications and Operational Specifications are considered source material and need not appear in this bibliography. (Comment Card.)

3.4.3.3.2 Operating Instruction Cards. Operating instruction cards shall contain the following information (all are Comment Cards):

(a) A Testing Routine description which describes the generation of data in the subroutines and gives enough information about the testing routine to delineate the parts of the subprogram

that it tested. Sample input and output examples may be given to help clarify the functions of the subprogram. These examples may be taken from the examples used to document the Testing Routine for this subprogram.

(b) A list of auxiliary equipment required by this subprogram, with the type, model number, and unit designation, if required. If manual page references for this equipment would be helpful in running the program, the manual name and edition should also be given.

(c) A list of sense switches which must be set by the operator at the beginning of the program or at an intermediate point during execution, along with their settings.

(d) A list of error possibilities which require operator intervention. Further, if some errors cause termination of the run, these errors should also be listed here. The information required in the Comment Cards is:

(1) Type of error, its display or diagnostic characteristics,

(2) Possible locations in the subprogram,

(3) Operator (or programmer) intervention to bypass or correct this error.

3.4.3.3.3 Data Unit Description Cards. Local data design units must be documented by preceding each definition with comment cards which adequately describe the data's function in the subfunctional program.

3.4.3.4 Internal Comment and Flow Chart Cards. Internal Comment Cards describe the coding techniques used in coding a particular step in a subfunctional program grouping within a subroutine. They are mainly a debugging aid.

3.4.3.4.1 Flow Chart Cards specify the programming and algebraic steps taken to achieve the subfunctional result required. Flow Chart Comment Cards describe in engineering English the programming steps documented by the Flow Chart Cards. The number of the functional grouping on the logic flow block diagram which contains

the logic being coded by these instructions or steps should appear on the first flow chart card of such a functional grouping.

3.4.3.4.2 Internal comment cards provide the following information for each step in the coding.

(a) The card numbers of any transfer points which transfer into the beginning of this step.

(b) List of maximum and minimum time estimates in machine cycles if the program is for a real time problem.

(c) If Brief Groups of Coding (see 3.3.3.2) are used, the name of the block of coding and its author should be referenced.

(d) Any use of packed data, linked lists, or second-level indexing must be referenced by an example. The bit configuration of packed data must be given. This need only be done before the first use of the word, list, or array in any subfunction. Its use in subsequent steps must be referenced to point back to its first use.

(e) The card number of any instruction written on by this instruction.

(f) Any instruction which is written on at execution time must be accompanied by a comment card giving the card number(s) of the instruction(s) which wrote on it.

(g) All steps in coding which would not be understood by a programmer with three months' coding experience must be explained by comment cards.

3.4.3.4.3 Flow Chart Cards will bear a one to one correspondence to the blocks of the Functional Flow Chart as described in 3.1.5.4. Thus, before each block of programming which performs a sub-subfunction as indicated by one block of the Functional Flow Chart for this subfunction, the programmer must place an algebraic or programming oriented description of the step to be done. Symbolic names are to be used, and a uniform assignment of verbs to correspond to the Flow Chart symbols used within the block will make these Flow Chart Cards more understandable.

3.4.3.4.4 Flow Chart Comment Cards will precede the Flow Chart Cards for a sub-subfunctional block of programming and will correspond exactly to the Program Flow Diagram entry for this block of programming.

3.4.3.5 Final Comment Cards

3.4.3.5.1 Indexes. If the compiler used does not produce the indexes listed as "required" under 3.6 (program assembly listings), then the final comment cards must reproduce these indexes.

3.4.3.5.2 Groups of operations which were repeated more than once in the routine should be given a name, described, and a list of the sections of the routine where they were used should be given. The author and the senior analyst who checked the coding should be given for each such group.

3.4.3.5.3 Suggested Revision Digest. Because of the dynamic nature of military tactical situations, revisions are anticipated. As such, a repository for potential areas of program improvement is provided. All suggestions in these cards will be reviewed by FCPC for possible incorporation concurrent with the subsequent revision. Each suggestion must contain the name of the programmer suggesting the revision; date the Comment Card was written; part of the program (by Flow Chart number) the suggestion pertains to; the suggestion; and concurring programmers.

NOTE

Where Testing Routine Comment Cards differ from the above Comment Cards, they will be described in the section on testing routines.

3.5 PROGRAM CARD DECK

3.5.1 The program card deck is the collection of IBM cards on which the source language coding and comment cards have been keypunched. All cards shall be interpreted in a plain-language character set. One program card deck shall be prepared for each program.

3.5.2 Color and Spacing

3.5.2.1 Program cards shall be white with cut corners. Spacing will be consistent with compiler language requirements. Comment cards shall be buff with square-cut corners. Header cards should be bright colored or have a bright, contrasting top edge.

3.5.3 Type

3.5.3.1 Cards shall be standard 80-column IBM cards.

3.5.4 Summary

3.5.4.1 The use of color is limited to that which will serve a functional purpose, such as enabling FCPC personnel to locate rapidly the cards associated with routines and subroutines within the deck.

3.6 PROGRAM COMPILATION LISTING

3.6.1 A program compilation listing is a computer generated listing of the compiled source program with its comment cards and its translation into the object program in core memory. In addition, some compilers provide symbolic reference tables, literal reference tables, etc. The following lists should either be compilation output or if the compiler will not provide them, final comment cards (see 3.4.3.5).

3.6.2 Date of Compilation

3.6.2.1 An alphabetized list of all the labels used in the program, their assigned core location and the core locations which refer to them.

3.6.2.2 A list of all the labels used in the program arranged in increasing order of their core location assignment, or a list of the labels and their compiler generated or sequence number arranged in card sequence number order.

3.6.2.3 A list of variable names (or table and field names) arranged in alphabetic order with their use referenced by core location or sequence number.

3.6.2.4 A list of index register usage should be supplied. Ideally, this should provide the index, the index name used, the procedure it was used in and the memory locations which referred to it.

3.6.2.5 A separate list for each procedure should give the number of inputs, outputs and abnormal exits for each procedure; it should also list the other procedures used by this procedure and the procedures which use this procedure. Any special switches, indices, labels or variables used by the procedure should appear in this report. The amount of memory occupied by the procedure should be given too.

3.7 TESTING ROUTINE

3.7.1 A testing routine is a computer program which tests another computer program. It reads in or generates input data, calls the program to be tested (as a subprogram), and analyzes the output from the tested program. The testing routine output consists of timing information, storage use, and any programming errors discovered in the tested program.

3.7.2 Requirements

3.7.2.1 The programmer of a testing routine shall code the testing routine from the Operational Specifications and the Program Flow Diagram of the program to be tested. If the Operational Specifications place limits on the timing and the amount of data which the tested program must process, then the testing routine will exercise the tested program to these limits. A testing routine shall not be written by the programmer of the tested program. Testing routines are considered to be a deliverable program routine; therefore, all requirements for Program Flow Diagrams, Functional Descriptions, Functional Specifications, Functional Flow Charts, program card deck and source program listings shall apply to Testing Routines.

3.7.2.2 Testing Routines are divided into two types:

(a) Those which test each branch point possible within a program and which test the program for extreme cases of storage use and timing limits.

(b) Those which supply simulated data which represent the ordinary use of the tested program. This type of testing routine gives timing and use statistics which reflect its use when the tested program is operational.

3.7.2.3 The first type of testing routine shall be called a "Logical Routine", the second type shall be called a "Simulation Routine". They are described in detail in 3.7.3 and 3.7.4. Comment Cards for both types of testing routines are described in 3.7.1.3. If the system for which the programs were written have a module test tool which will check the timing of modules in the total program, then the module test tool may be a part of the Simulation Routine and the Logical Testing Routine.

3.7.3 Logical Testing Routine

3.7.3.1 Data Design. The Logical Testing Routine programmer shall design input data for the testing routine which cause the flow of the program, during execution, to enter every possible combination of paths and to evaluate the limits set by the Operational Specifications. Thus, if it is possible for the flow of the program to enter branch B after branch C, one set of data read by the testing routine will cause this sequence of branches to be executed. Further, if the Operational Specification requires, for example, that up to 100 tracks be processed at one time, then one set of data read by the testing routine will assume that 100 tracks are processed by the tested program.

3.7.3.2 Data Output. Output will consist of a description of the combinations of branches taken, the timing required to execute each of these combinations of branches, the output resulting from each of these executions (a core display is not necessary, only a printout or display of the values of those variables which would be input

to a subsequent program or cause further branching to a subsequent program), any error conditions encountered, and the absolute core location where they were encountered.

3.7.3.3 Logical Testing Routine for Executive Control Routine. The Logical Testing Routine for an Executive Control Routine (which integrates the operations of several programs) should be capable of calling for all logically possible combinations of these programs. It will thus have the ability of calling for specific combinations of programs with data designed to exercise the programs according to Logical Testing Routine requirements.

3.7.4 Simulation Testing Routines

3.7.4.1 Inputs. Inputs shall be as statistically close to input data of real performance as possible. If it is feasible, the auxiliary equipment which supplies the input data in the "real world" use of the program should be used to generate input data. If government prepared data is available which meets the requirements of the Operational Specifications, it should be used as at least part of the input data for the Simulation Testing Routines. Since the results of Simulation Testing Routine runs will be used to evaluate the effectiveness of the tested program, data as similar to real data as possible should be prepared. It may be possible, in systems programming (where many small programs fit together to form a system), to use the output from a Simulation Routine run of one program as input to another program within the system (for amplification see 3.7.4.3).

3.7.4.2 Outputs. Output will be divided into five types:

(a) A statistical evaluation of the input data and a breakdown as to the types of input entering the program.

(b) Total program processing time for a given sample of the input data and the percentage of use the different steps (in the subfunctional breakdown of the program) of the program received for a given sample of the data.

(c) A statistical evaluation of the output from the program.

(d) Error diagnostics and the conditions under which they occurred, including a sample of the data inducing the error.

(e) Storage usage during execution (consisting of the maximum amount of storage use for different tables, the least amount of storage used, and the average used), sampled over a wide enough range of times to provide valid information.

3.7.4.3 Simulation Testing Routines for Executive Control Routines. A Simulation Testing Routine for an Executive Control Routine should be capable of displaying the integration of all logically possible subprograms within the system. The requirements under 3.7.4.1 and 3.7.4.2 shall hold for this testing routine. In addition, frequency of use figures for each of the programs (as in the steps of the subfunctional breakdown in 3.7.4.2 (b)) should be part of the output. It should be possible to select the sequence of combinations of programs as part of this testing routine.

3.7.5 Comment Cards for Testing Routines

3.7.5.1 The Comment Cards of a testing routine differ somewhat from the Comment Cards for a program as described in Section 3.4.

3.7.5.2 Lead cards precede any instructions in the testing routine. They consist of the two types listed in 3.4.3.2. They need only give the following:

(a) Which of the two types of testing routine this program is, and the name of the tested program with its configuration control number.

(b) The major steps within the testing routine need not conform to configuration control step numbers for the tested program. The identification of the testing routine is deemed sufficient.

3.7.5.3 Operating instructions in the lead cards will be more specific than the program comment cards on input and output.

(a) Input and output requirements should be illustrated in the Comment Cards by examples of the bit configuration (where packed words are input) the quantities to be expected, and the limits in

size of individual numbers (where full words are input). Expected output should be similarly illustrated. The kind of equipment supplying this data should be given here. This is not auxiliary equipment, but rather it is the source of the input. For example, if magnetic tapes supply the input, the density of the tape, the type of unit which generated the tape, whether the tape is in the binary or BCD mode should all be given in the Comment Cards. If the testing routine requires special system data cards, these cards and their function shall be described in the lead comment cards. The remainder of the operating instructions, Comment Cards (b) through (e), hold as described in 3.4.3.3.2.

3.7.5.4 Subfunctional Lead Cards. Subfunctional lead cards for testing routines are rarely relevant, since most testing routines are quite short and depend on their data, whether read in or generated, to exercise the tested program.

3.7.5.5 Internal Comment Cards. Internal comment cards (3.4.3.4) hold for testing routines where the coding is sufficiently sophisticated to merit them. It is not expected that as much debugging and revision of testing routines will be needed as compared with the tested program, so internal comment cards will not be as necessary for the maintenance of the testing routine. Flow Chart Cards and Flow Chart Comment Cards will reflect the Functional Flow Chart blocks and Program Flow Diagram blocks, respectively.

3.7.5.6 Final Comment Cards. Final comment cards may be deleted for testing routines.

3.8 FORMAT

3.8.1 The following paragraphs define the format to be used for the preparation of each of the deliverable end items required by the specification.

3.8.2 Program Flow Diagrams

3.8.2.1 Original art shall be prepared on a translucent material such as vellum, with 1/8" dropout blue quadrille markings. The

material shall be selected to be compatible with ozalid or equal process. Image area shall be 20" high by length as required, but not to exceed 54". The image area has been established to be compatible with 2:1 reduction for final production.

3.8.2.2 Lettering. Lettering typewritten on translucent friskit and affixed to the diagram is preferred. Since this technique has not been accepted as an industry standard, legible hand lettering is acceptable.

3.8.2.3 Inking. Line art shall be inked according to the following line weight list.

Action or Decision Boxes	Leroy No. 3 or equal
Program Flow Lines	Leroy No. 5 or equal
Grouping Lines	Leroy No. 1 or equal

Major routines shall be enclosed by a solid line. Subordinate routines within the major routine shall be enclosed by a dashed line with 1/2" dashes with 1/4" spacing. The second level of subordination within the major routine shall be enclosed by dashed line with 1/4" dashes with 1/8" spacing. The third level of subordination within the major routine shall be enclosed by dashed lines with 1/8" dashes with 1/8" spacing. The engineering data block shall be located in the lower right-hand corner, inside the image area.

3.8.2.4 Titles. Drawing titles shall be located in the bottom margin outside the image area such that they end flush right with the image area. Figure numbers are not required. The engineering data block and the prime number identifier at the functional block provides adequate identification.

3.8.2.5 Review Copies. Review copies shall be ozalid process or equivalent reproductions of the original art. Because of the short life span of the review diagram, low cost is the prime criterion.

3.8.3 Functional Descriptions of Logic Flow

3.8.3.1 Original manuscript will be typewritten on vellum to facilitate ozalid or equivalent reproduction. Orange backing is not required but is acceptable. Manuscript shall be single spaced. A 3/4" margin (approximately) shall be maintained on all edges.

3.8.4 Functional Specification

3.8.4.1 Upon completion of the design review and incorporation of all changes and comments resulting from design review, the Program Flow Diagrams and the functional description shall be combined and renamed Functional Specification. Identifying numbers shall be provided by the contracting agency.

3.8.4.2 Final Copies. Final copies shall be ozalid or equivalent reproductions of the text of the functional descriptions. Right-hand margins shall be unjustified. Proportionate spacing typewriters with a 12-point sans serif demibold type face are preferred; however, clean, legible, type from any typewriter is acceptable. The Program Flow Diagrams shall be inserted immediately following the corresponding sections of text. The diagrams in the Functional Specification shall be ozalid or equivalent reproductions of reduced film (paper or acetate) copies of the Program Flow Diagram. Two-to-one reduction provides the transition from the original art size to 11" x length fold-out size which is compatible with the text pages.

3.8.5 Listings

3.8.5.1 To be discussed.

3.8.6 Functional Flow Charts

3.8.6.1 Since Functional Flow Charts may be prepared by either of two techniques the format will depend on the technique chosen.

3.8.6.2 Machine Prepared Functional Flow Charts. Machine prepared Functional Flow Charts shall be offset reproductions of the computer

printout. The machine printout shall be photoreduced to 50 percent. The paper stock shall be 80-pound bond; binding shall be in accordance with 3.8.7.

3.8.6.3 Manually Prepared Functional Flow Charts. Manually prepared Functional Flow Charts shall be offset reproductions of the photoreduced charts. Paper stock shall be 80-pound bond. Binding shall be in accordance with 3.8.7.

3.8.7 Binding

3.8.7.1 Functional Descriptions of Logic Flow and Functional Specification shall be prepared in looseleaf form to facilitate the insertion of replacement pages. For manuals of less than 1/2-inch thickness, commercial metal fasteners are to be used. The pages shall be punched or drilled as follows (all eimensions in inches):

Hole-type binders	8-1/4 x 10-3/4
Number of holes	3
Hole size	1/4
Distance, center to center	4
Distance to binding edge	7/16

3.8.7.2 For manuals that exceed approximately 1/2-inch thickness, looseleaf mechanical opening three-ring binding shall be used. Punching or drilling of revision pages shall be the same as for the original manuals.

3.8.8 Paper and Cover Stock

3.8.8.1 Any good-quality paper stock which is suitable for the intended method of reproduction will be satisfactory for text and fold-in pages. Cover stock satisfactory for the intended use is acceptable; however, the color shall conform to security requirements.

3.8.9 Security Classification

3.8.9.1 The security classification of a drawing shall be as designated by the bureau or agency concerned. The Security

Requirements Check List (DD Form 254), which constitutes a part of the contract for all classified material, identifies and indicates the classified features. All pages of classified manuals and supplements shall be marked in accordance with the Industrial Security Manual for Safeguarding Classified Information.

3.8.9.2 Additional Security Markings. When a manual contains information of a higher classification than that of the equipment it concerns, the appropriate classification of all classified data contained within that manual shall be identified by a classification letter(s) enclosed in parentheses and positioned as follows:

(a) At the beginning and end of paragraphs and subparagraphs.
(b) At the end of the subject or title.

3.8.9.3 Classification Letters. The classification letters assigned to the various levels are: (TS) Top Secret, (S) Secret, (C) Confidential, (CMH) Confidential – Modified Handling Authorized, (CRD) Confidential – Restricted Data, and (NOFORN) No Foreign Nationals. Although it is not intended that each and every item of information bear a classification letter, the letter (U) shall be used as directed by the bureau or agency concerned to denote unclassified data.

3.8.10 Notes, Cautions, and Warnings

3.8.10.1 Notes, cautions, and warnings shall be used to emphasize important and critical instructions, consistent with the need. Notes, cautions, and warnings shall immediately precede the applicable instructions, and shall be selected in accordance with the following:

(a) "NOTE" - Concerns an operating procedure or condition which should be highlighted.

(b) "CAUTION" - Concerns an operating procedure or practice, which if not strictly observed, will result in damage to or destruction of equipment.

(c) "WARNING" - Concerns an operation procedure or practice which, if not strictly observed, will result in injury to personnel or loss of life.

3.8.11 Numbering and Identification

3.8.11.1 The subordinated numbering of the Program Flow Diagrams shall be the basis for numbering all associated documents. The Program Flow Diagram for the complete program shall be identified as 0.0. Each major routine identified on this diagram shall be numbered consecutively beginning with 1. The Program Flow Diagram for each of the routines illustrated on the complete program diagram shall carry the identifying number provided on the Program Flow Diagram for the computer program. Subroutines illustrated on these diagrams shall be numbered beginning with 1.

3.8.11.2 The Program Flow Diagram for each subroutine shall carry the same type of identifying number. Subordinate groupings within the diagram shall be numbered according to the same indenture scheme.

3.8.11.3 The Functional Description and Functional Specification paragraph numbering system shall provide a one-to-one reference between the Program Flow Diagrams and the associated text.

4. QUALITY ASSURANCE

4.1 Accuracy and completeness are the criteria for acceptance of computer program documentation; format is a consideration. It is presumed that quality assurance is an inherent part of all contracting organizations. The responsibility of FCPC is to check the effectiveness of the quality control organization at the contractor facility, rather than to be responsible for the quality of the end product. The Quality Assurance Monitor Policy described in this section provides guidelines to the contractor for quality control programs to assure end products of acceptable quality.

4.2 FCPC will implement quality control insurance checks by performing a detailed quality control inspection on a random sample basis, as described in the following paragraphs. All documents of the same type (i.e., program flow diagrams, listings, etc.) will be deemed to be equally acceptable or unacceptable as the sample taken.

4.3 OPERATIONAL SPECIFICATION – PROGRAM FLOW BLOCK DIAGRAMS

4.3.1 A check will be performed to ensure that the requirements of the Operational Specifications are satisfied. Logic Flow Block Diagrams will also be checked for adequate and logical subfunctional groupings.

4.4 PROGRAM FLOW BLOCK DIAGRAM – FUNCTIONAL SPECIFICATION

4.4.1 The Functional Description will be checked against the block diagram to ensure compatibility of numbering, text coverage of each block, and the technical adequacy of the paragraph describing the related block on the Program Flow Diagram.

4.5 FUNCTIONAL SPECIFICATION – PROGRAM ASSEMBLY LISTING

4.5.1 A check will be made of the adequacy and accuracy of the comments placed in the listing resulting from the comment cards developed during the programming process. The actual programming technique shall be checked for good programming practice.

4.6 FUNCTIONAL FLOW CHART DIAGRAMS — TESTING ROUTINE

4.6.1 The testing routine shall be checked against the Functional Flow Charts to ensure that the logic is exercised completely and that the testing routine does test the performance of the subroutine, routine, or program.

4.7 The objective of quality control is to provide near-perfect documentation, not to criticiae human frailty. Discrepancies found by FCPC personnel will be graded into three categories as follows:

(a) Critical - Mandatory correction
(b) Secondary - Improvement to be made next revision
(c) Minor - Correction to be made when page is revised (typo, misspelling)

To preclude double retyping to incorporate both FCPC and contractor comments, pencil-marked copy may be submitted by the contractor to FCPC for final quality control inspection.

5. PREPARATION FOR DELIVERY

5.1 PACKAGING AND PACKING

5.1.1 All items shall be packed to preclude the possibility of damage in transit. Multivolume manuals shall be furnished as complete sets except for the manuals shipped for stock. Stock copies of identical volumes shall be packed and shipped in common container(s).

5.1.2 Bulk Shipment

5.1.2.1 Data items shipped in bulk shall not be individually wrapped. Containers shall comply with the Uniform Freight Classification Rules, or other regulations applicable to the mode of transportation.

5.1.3 Marking

5.1.3.1 On bulk shipments, interior packages and exterior shipping containers shall be marked with the following information for each item enclosed.

(a) Box (number) of (number)--to be shown on each container for multiple container shipments.

(b) Publication number.

(c) Quantity (in package).

(d) Contract or order number.

5.1.3.2 The words "FOR STOCK" shall be marked on the package or packages destined for stock. The publication numbers shall be indicated on the shipping documents.

6. NOTES

6.1 ORDERING DATA

6.1.1 Procurement documents shall specify the following:

(a) Title and number of specification.

(b) Security classification.

(c) Quantity of manuals (revisions of APR pages), supplements and preventative maintenance data required.

6.2 LIABILITY NOTICE

6.2.1 When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not be be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.